design ideas

Edited by Bill Travis and Anne Watson Swager

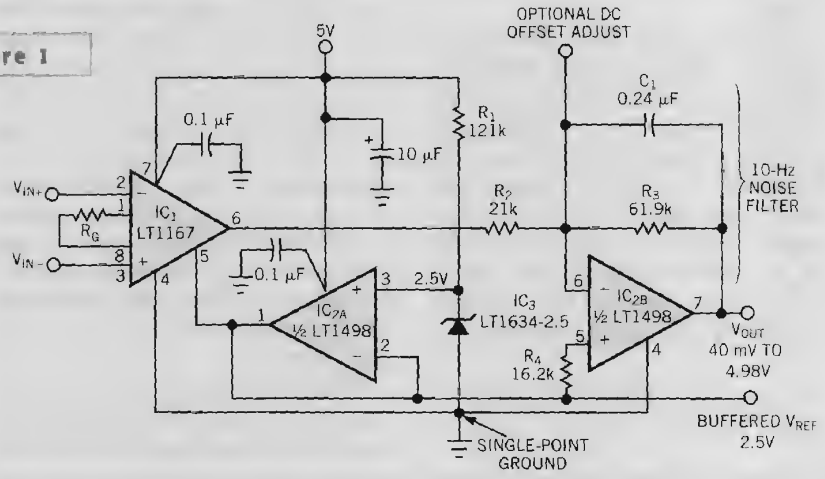
Instrumentation amp works from one supply

Adolpho Garcia, Linear Technology Corp, Milpitas, CA

Many single-supply applications need precision amplifiers that can operate from 5V or lower. Although many precision, single-supply op amps are currently available for configuring as 2- and 3-amplifier instrumentation amplifiers (IAs), these designs require great attention to detail to achieve accuracy and precision. Furthermore, although single-supply IA ICs are available, these products trade off dc and ac performance for low-supply-voltage and -current operation. Dual-supply IAs still offer the best performance.

Achieving high-precision performance in a single-supply application is practical, because the majority of sensor applications provide an output signal centered about the midpoint of a circuit's supply or reference voltage. Examples include strain gauges, load cells, and pressure transducers. In these applications, the signal-conditioning circuitry is not required to operate near the sensor's or circuit's positive supply voltage or ground. Even though the signal-conditioning circuitry need not operate at the extremes of the input-voltage range, the output-voltage swing of the circuitry should be as large as possible to achieve maximum dynamic range. The circuit in **Figure 1** achieves high-precision performance while operating from a 5V supply.

Figure 1



A 2.5V reference IC provides a stable supply midpoint to configure a single-supply instrumentation amplifier.

The trick here is to reference the dual-supply IA's inputs to a stable supply midpoint, then follow the IA with a single-supply precision op amp with a rail-to-rail output swing. This "composite" IA uses IC_1, an LT1167 high-performance IA, for the input stage, and IC_2, an LT1498 high-speed, rail-to-rail input/output dual op amp for the output stage. IC_3, an LT1634 micropower 2.5V precision shunt reference, provides a stable 5V-supply midpoint. The output of IC_3 connects to the input of IC_{2A}, configured as a voltage follower. The output of IC_{2A} provides a low-impedance source for IC_1's reference pin 5, which exhibits 20-kΩ input resistance and input current to 50 µA maximum. A low-impedance source is necessary to maintain IC_1's high common-mode rejection. In addition, IC_{2A}'s output stage can provide load currents to 20 mA for additional external circuitry without affecting IC_3's accuracy.

IC_{2B} is a gain-of-3 inverter whose out-

TABLE 1—SUMMARY OF STATIC AND DYNAMIC CHARACTERISTICS

Circuit gain	RG (Ω)	V*OS (µV)	Bandwidth (kHz)	0.1- to 10-Hz noise* (µV p-p)
10	20.5k	1300	900	2
30	5.36k	450	850	0.7
100	1.5k	160	500	0.4
300	487	100	160	0.3
1000	147	90	40	0.3

*** Referred to input**

put can swing ±2.5V (rail-to-rail) with only ±0.82V drive from IC_1. The primary reason for choosing an inverting-amplifier configuration for the output stage is to make system dc-offset adjustments available. You can connect trim networks to the inverting terminal of IC_{2B} without affecting the static or dynamic behavior of the circuit. However, you should design the trim range so as to not adversely affect the output dynamic range of the circuit.

IC_1 maintains its high-linearity performance with a 5V supply because its front end is configured to operate from dual supplies, and the circuit in **Figure 1** relaxes its output drive level. Because IC_3 level-shifts the entire circuit above ground, you measure the circuit's final output voltage with respect to 2.5V, not 0V. An expression for the gain of this composite IA combines the gain equations of IC_1 and the gain-of-3 inverter:

$$\text{GAIN} = \left[1 + \left(\frac{49.4\,\text{k}\Omega}{R_G}\right)\right] \times \left(\frac{R_3}{R_2}\right)$$

As shown in **Figure 1**, choosing R_G = 1.5 kΩ yields a gain-of-100 composite configuration. You can obtain other gain values with different values of R_G, as shown in **Table 1**. Even though it's not necessary that the inputs to the circuit operate at the positive rail or ground, wide input common-mode operation is always beneficial.

In this configuration, IC_1's input stage can accept signals as high as 3.7V (common mode plus differential mode) with no loss of precision. In fact, at low circuit gains, the circuit's common-mode input-voltage range spans 2.25 to 3.45V. This wide common-mode range allows room for the full-scale differential input voltage to drive the output ±2.5V about the reference point (V_{REF}). Another application hint regarding this circuit: Though IC_1's input bias currents are lower than 1 nA, the circuit's differential-input terminals must have a dc return path to the power supply.

Table 1 summarizes the static and dynamic performance of the composite IA. Nonlinearity for all gain values is lower than 0.006%. The transient response of the circuit as a function of gain and load is well-behaved and is attributable to IC_2's wideband rail-to-rail output stage. Note that measurements of small/large-signal transient response and circuit bandwidth reflect the absence of C_1. The circuit's 10-MHz gain-bandwidth product and 6V/µsec slew rate ensure that the small-signal performance is primarily a function of IC_1's characteristics. Capacitor C_1 is beneficial in low-frequency applications (signal bandwidth lower than 20 Hz), to eliminate or significantly reduce noise pickup. Noise can also sneak into the circuit via the input terminals of IC_1, especially if the sensor is located some distance from the signal-conditioning circuitry. This type of noise can cause a shift in the input offset voltage of IC_1, thereby producing errors. This effect is commonly termed RF rectification. You can easily add a differential filter to IC_1's input terminals to reduce this effect. (DI #2498).

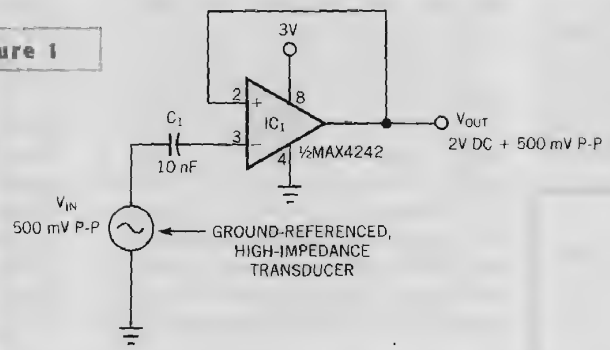
To Vote For This Design,
Circle No. 301

Amplifier requires no dc bias

John Guy, Maxim Integrated Products, Sunnyvale, CA

Intrinsically capacitive transducers and other high-impedance signal sources usually require ac coupling and a buffer amplifier to condition the signal for further processing. Buffers take many forms, but most of them compromise signal quality through the use of external resistors that provide a dc path for the input bias current. Recent improvements in op-amp technology allow ac-coupled inputs without the need for bias resistors. The new op amps have inputs that operate within the supply-voltage span. Some are CMOS, but many use complementary bipolar transistors. For the latter, each input connects to both a npn-difference pair and a pnp-difference pair. Combining these parallel input stages ensures that the second stage has a wide input common-mode range. Thus, some op amps exhibit an input-voltage level for which bias current to the npn pair exactly equals that from the opposite-polarity pnp pair, causing the input bias current to go to zero (**Figure 1**).

Figure 1



A self-biasing amplifier buffers a high-impedance signal without the need for bias resistors.

For the amplifier shown, this bias-canceling effect self-biases the inputs at a level consistently close to $2V_{CC}/3$. To avoid the distortion effects that increase with signal deviations from $2V_{CC}/3$, you should limit the input signal swings to 500 mV p-p. If necessary, downstream circuitry can remove the dc offset while providing gain and filtering. Even with small, low-cost coupling capacitors, this circuit's ultrahigh input impedance and absence of bias resistors allow operation at frequencies well below 1 Hz. (DI #2496).

To Vote For This Design,
Circle No. 302

Halogen light dimmer provides infinite control

Suded Emmanuel, Emmanuel's Controls, Auckland, New Zealand

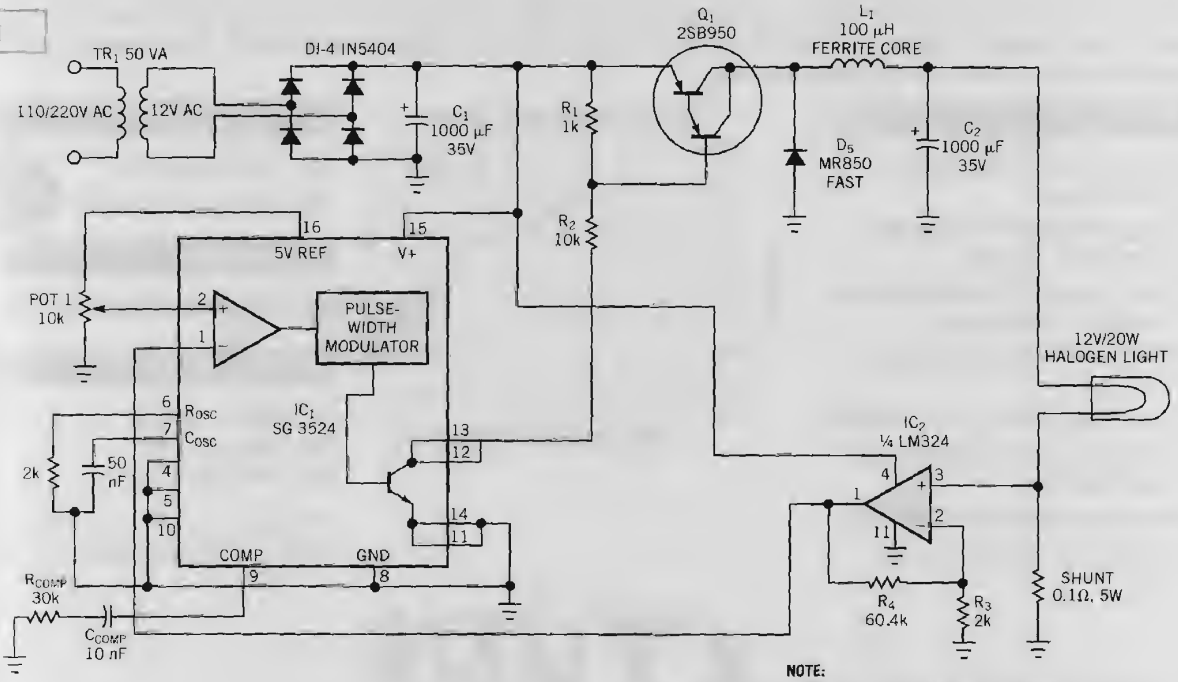
Modern lighting systems use halogen lamps, most of which run on 12V ac from a transformer. The dimmer circuit in **Figure 1** can change the intensity of the light from zero to maximum. The dimmer operates at approximately 12V, unlike the usual ones that function by adjusting the firing angle of the 110 or 220V mains supply.

The dimmer works to inject a constant current into the halogen lamp and to regulate that current using pulse-width modulation (PWM) according to a potentiometer-controlled input, or a 0 to 5V signal, or even an analog output from a µC. 12V ac from the transformer, converted to 16.8V dc, powers the SG3524 PWM circuit (IC_1). An RC circuit sets the approximately 10-kHz operating frequency. The output of the PWM IC drives the power transistor (Q_1), a pnp Darlington. The collector of Q_1 connects to a 10-turn ferrite-core inductor; a 1000-µF capacitor affords filtering to provide the bulb with dc current. Op-amp IC_2 amplifies the drop across the shunt resistor and feeds the amplified signal back to IC_1. IC_1 compares the feedback signal with the desired input level from the potentiometer, 0 to 5V, then controls and regulates the current in the bulb.

Current regulation is important, not only because it makes dimming possible, but also because it protects the bulb at start-up (when the bulb is cold). The constant current gives the filament longer life and makes the bulb immune to line-voltage disturbances. Some designs use "electronic transformers," which are basically switching power supplies that drop the mains voltage from 110/220V ac to 12V dc that's pulsed at high frequency. These systems generate higher RFI than the design in **Figure 1**. In this design, because the controlled variable is current, not voltage, you could use supplies higher than 12V ac to compensate for the drop in the connecting wires in case you wish to place the halogen bulb and dimmer at some distance from the transformer. (DI #2497).

To Vote For This Design, Circle No. 303

Figure 1



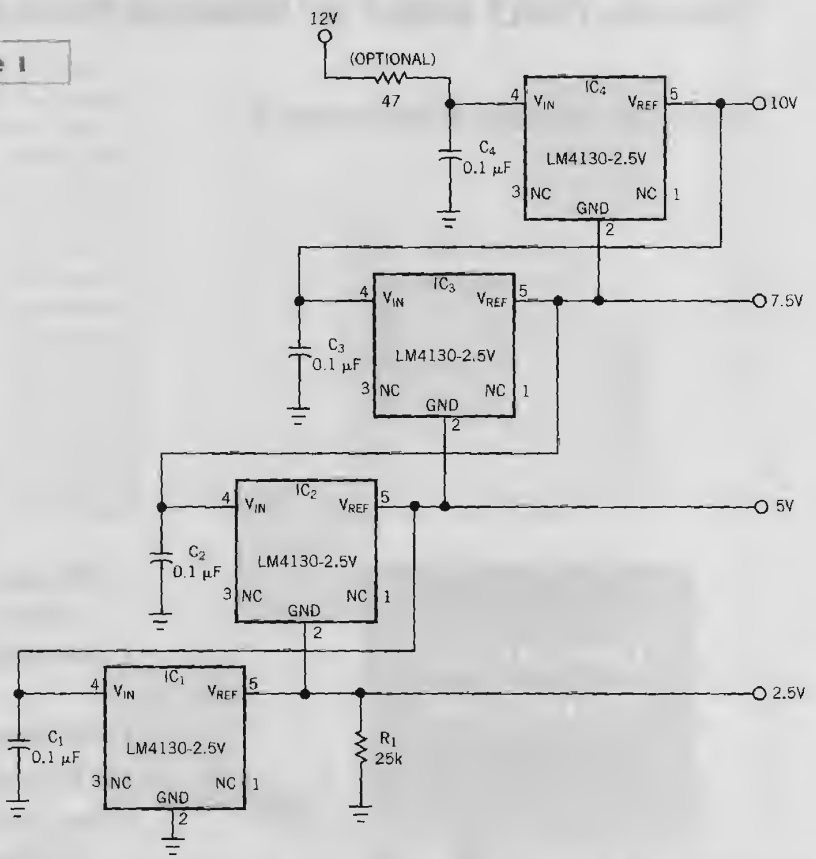
NOTE:
FOR A 20W LIGHT, MAXIMUM CURRENT IS 1.6A; THAT IS, A 0.16V DROP ACROSS THE SHUNT. THIS DROP EQUALS 5V INPUT TO THE PWM IC. THE GAIN OF THE NONINVERTING OP AMP IS 31.2.

Current, not voltage, controls halogen-lamp dimming in this simple scheme.

Cascoded stack yields multiple voltages

Clayton Grantham, National Semiconductor, Tucson, AZ

Figure 1



If you understand its limitations, this circuit provides an easy way to obtain accurate multiple voltages.

ALTHOUGH THE CASCODE voltage-reference configuration in **Figure 1** may seem obvious, the choice of R_1 and the bypass capacitors is critical. At first glance, stacking up references of the same voltage to produce a collection of voltages seems straightforward. However, nothing comes without precautions and an understanding of limitations.

As a case in point, the CMOS references used in this circuit have a 5.5V supply-voltage limitation. The circuit operates at 12V, which exceeds the limitation; thus each of the individual references must proportionally scale down the 12V input. IC_2's 5V output powers IC_1, IC_3's 7.5V output powers IC_2, and IC_4's 10V output powers IC_3. Similarly, the references take their ground potentials from references beneath them in the cascode, to keep the total supply span for each IC below 5.5V.

The line-regulation errors of IC_1, IC_2, and IC_3 are near perfect, because their individual V_{IN} potentials come from a solid voltage source. The input source of 12V has a range of 10.2 to 14V, but if it dropped below 10.2V, each output voltage would be accurate until the input source dropped to within 200 mV above each stack voltage. For example, if the input source were at 5.2V, the 2.5 and 5V outputs would be within specification and the 7.5 and 10V outputs would be close to 5.2V. In this way, as a 12V battery collapses, external circuitry dependent on the lower voltages would still be functional.

Let's examine R_1 and its limits. The LM4130 sources current very well (to 20 mA), but it sinks only 10 µA. Thus, R_1 is a resistive pull-down for IC_2's quiescent current (50 µA). R_1 must be a maximum of 25 kΩ to keep IC_2 biased for worst-case specs over temperature. This 100-µA bias current also keeps IC_3 and IC_4 biased. The ICs of the stack roughly share a single quiescent current, as opposed to a parallel configuration that would draw four times the quiescent current. It is also true that each output source current has a ripple effect from previous ICs in the stack. Thus, cumulatively, the voltage outputs (2.5, 5, 7.5, and 10V) can source as much as 20 mA (5 mA from each output). Load currents of the lower references have effects on the voltages of the ICs stacked above. If the output impedance (0.075Ω) of the LM4130 were not very small, then its effect would create a large crosstalk error, with one output causing another to vary. For example, with the 2.5V output loaded with 20 mA and accounting for the Z_{OUT} effects of both IC_2 and IC_3, the worst-case change in the 10V output is 3 mV, or 0.03%.

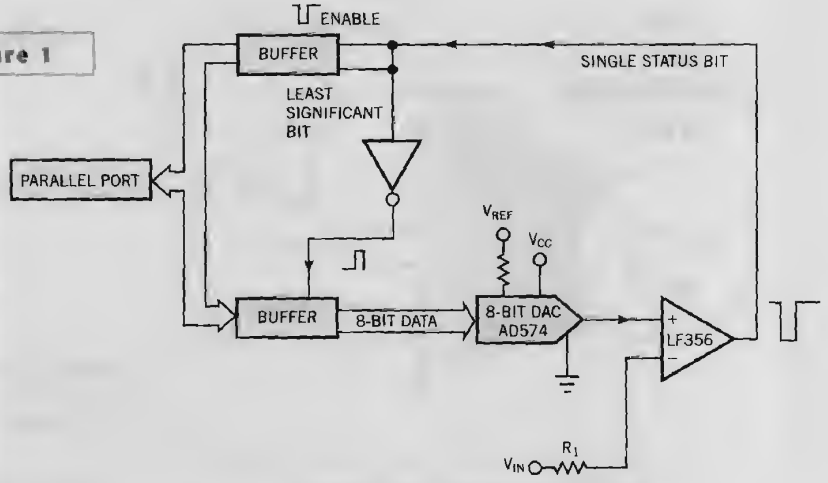
The bypass capacitors C_1 through C_4 have a secondary function, other than input bypassing, that overcomes another limitation of the cascode configuration. They compensate the internal LM4130 output. This output stage is a common-source PMOS FET with local feedback that reduces the output impedance beyond 100 kHz. Ceramic, tantalum, or aluminum-electrolytic capacitors work to keep the stack of voltage references from start-up instabilities and oscillations. Another limitation of the cascode arises if the lower voltages (2.5, 5, and 7.5V) become grounded. Any continuous short circuit would produce excessive power dissipation in the LM4130. The optional 47Ω series resistance in the 12V line protects the ICs under worst-case conditions. Note that despite the limitations, the accuracy of the stack voltages tracks the LM4130's accuracy. And, as with accuracy, the temperature-coefficient errors do not degrade up the stack. (DI #2501).

BIOS interrupt performs A/D conversion

J Jayapandian, IGCAR, Tamil Nadu, India

Most A/D-conversion techniques use dedicated hardware, usually a single-chip IC. The flexible conversion technique allows you to use successive-approximation, ramp-type, or other converters by writing the appropriate control software. The design in **Figure 1** uses a PC's parallel port for interface to a DAC and a special BIOS interrupt (INT 1CH) for the conversion process. The INT CH hardware interrupt is available in all PCs. It automatically occurs 18.2 times per second; the BIOS-timer interrupt invokes the interrupt after the interrupt updates the time-of-day count. The INT 1CH handler routine activates the hardware in **Figure 1**. The analog input V_{IN} connects to the inverting input of the LF356 op amp; the noninverting input connects to the output of the AD574 8-bit DAC. The output current of the DAC follows the digital pattern from the PC's port (you can use the LPT port for 8-bit applications).

Figure 1



You can use a PC's special BIOS interrupt to implement an A/D converter.

The interrupt-handler routine (**Listing 1**) for INT 1CH sets the required 8-bit digital image for the DAC's input and monitors the Status bit from the op amp's output. You can write the software for any conversion technique. Here, the INT 1CH handler routine, written in Turbo C, implements a counter-ramp conversion technique. The routine in **Listing 1** uses INT 1CH and the LPT2 port. The vari-

LISTING 1—HANDLER ROUTINE FOR A/D CONVERSION

```
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#define OUT_PORT  0X378   /* Out port address of LPT2 */
#define CTRL_PORT 0X37A   /* Control port address of LPT2 */
#define INTRTIMER 0x1C    /* BIOS Timer (INT 1CH)Interrupt */
/*-------------------------------------------------------*/
/*-----------------GLOBAL VARS---------------------------*/
static int ADC_value,STATUS;
static int TICKER;
int i = 0;
void interrupt (*timerhandler)();
void interrupt ADCHANDLER();
/*-------------------Handler routine for INT 1C----------*/
void interrupt ADCHANDLER()
{
        disable();
            ++TICKER;
            for(i=0; i<256; i++)
            {
            outportb(OUT_PORT,i);
            ADC_value = i;
            }
            STATUS = inportb(OUT_PORT) & 0x01;
           enable();
} /* END OF ADCHANDLER */

void INSTALLADCHANDLER()
{
        disable();
        timerhandler = getvect(INTRTIMER);
        setvect(INTRTIMER,ADCHANDLER);
        enable();
}

void CLEARADCHANDLER()
{
        disable();
        setvect(INTRTIMER,timerhandler);
*/
        enable();
}

void main(void)
{
        clrscr();
        outportb(CTRL_PORT,0x01);
            INSTALLADCHANDLER();
            while (STATUS != 0x01);
            printf("ADC value is %x\n",ADC_value);
            CLEARADCHANDLER();
          //getch();          /* For testing this can be included */
        return;
}
```

able Ticker recognizes the occurrence of INT 1CH. For every Ticker, the handler routine writes the incremental data from 0 to 255 to the LPT2 port, and thus to the 8-bit DAC. The handler checks for the zero-status bit by reading the LPT2 port. If it reads a high-to-low transition in the Status bit, the handler writes the final bit pattern (a value between 0 and 255) to the DAC. The op amp compares the value of V_{IN} with the DAC's output. When the output reaches V_{IN}, the op amp's output (status) becomes 0V, and the handler stops incrementing the bit pattern presented to the DAC.

This final digital value corresponds to the analog input V_{IN}. For each occurrence of INT 1CH (Ticker), the design completes a conversion cycle. Make sure that the handler routine does not exceed the time of occurrence of the interrupt. You can implement a successive-approximation converter in the same way with this design by writing an appropriate handler routine. You can download **Listing 1** from *EDN*'s Web site, www.ednmag.com. Click on "Search Databases" and then enter the Software Center to download the file for Design Idea #2495. (DI #2495).

Generate frequencies with arbitrary relationships

Sanjay Gupta, NIIT Ltd, New Delhi, India

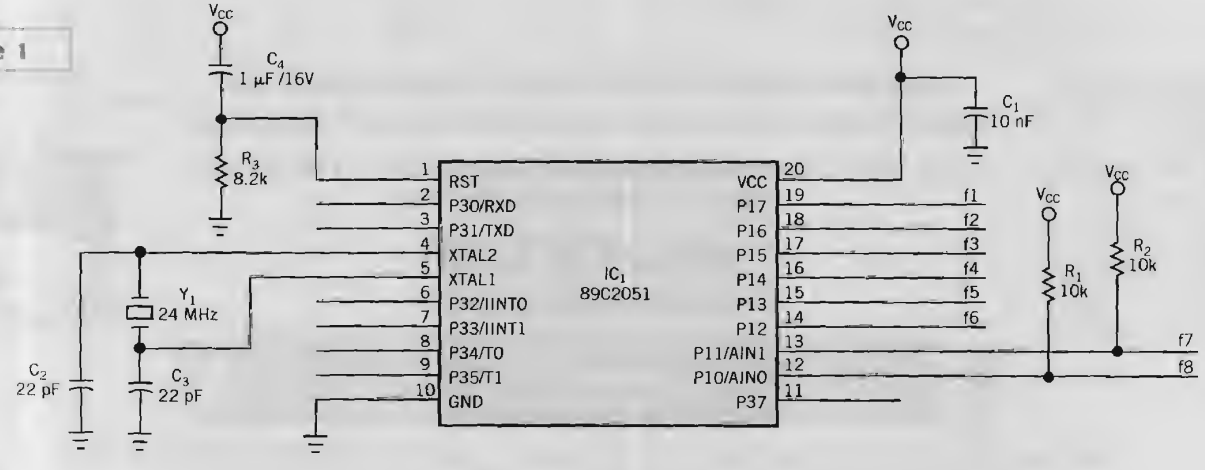
The circuit in **Figure 1** and the program in **Listing 1** use an Atmel 89C2051 µC to generate a variety of frequencies that have no specific relationship to each other. The example given here generates the following eight frequencies: 500, 700, 1000, 1050, 1100, 1500, 1700, and 2000 Hz. The program delivers these frequencies on eight I/O pins (P17 to P10, respectively) of the µC. The program associates each of these I/O pins with an internal-RAM address (f1 through f8).

The interrupt routine in **Listing 1** takes 492 oscillator periods to complete. At 24 MHz, this interval corresponds to 20.5 µsec. The internal timer T0 of the µC generates an interrupt every 25 µsec. During each interrupt cycle, each of the counts (f1 through f8) decrements by one. Whichever count decreases to zero, the corresponding output pin toggles and that count value reinitializes. Average action converter in the same way with this design by writing an appropriate handler routine. You can download **Listing 1** from *EDN*'s Web site, www.ednmag.com. Click on "Search Databases" and then enter the Software Center to download the file for Design Idea #2495. (DI #2495).

TABLE 1–PERIOD AND COUNT DATA FOR REPRESENTATIVE FREQUENCIES

Frequency	Period of half-cycle (µsec)	Closest half-cycle time period obtainable with 25-µsec interrupt interval	Error (%)	Count
500 Hz	1000	100	0	1000/25=40
700 Hz	714	700	1.96	700/25=28
1000 Hz	500	500	0	500/25=20
1050 Hz	476	475	0.21	475/25=19

Figure 1



You can generate arbitrary, unrelated frequencies using an 89C2051 µC.

curacy improves as the magnitude of the required output frequency decreases (**Table 1**). You can also obtain more accuracy by using the following techniques:

- Increase the processor-clock frequency, thereby reducing the time it takes to execute the interrupt routine.
- Reduce the number of clock cycles in the interrupt routine.

If you need to generate fewer frequencies, for example, the last six from the preceding list, then the time taken to complete the interrupt cycle reduces to 372 cycles (15.5 µsec). So you can program timer T0 to generate an interrupt every 18 µsec instead of 25 µsec. **Table 2** shows the new accuracy figures with fewer frequencies. You can download **Listing 1** from *EDN*'s Web site, www.ednmag.com. Click on "Search Databases" and then enter the Software Center to download the file for Design Idea #2500. (DI #2500).

TABLE 2–ACCURACY FIGURES FOR SIX FREQUENCIES

Frequency (Hz)	Period of half-cycle (µsec)	Closest half-cycle period obtainable with 18-µsec interrupt interval	Error (%)	Count
1000	500	504	0.8	504/18=28
1050	476	468	1.68	468/18=26
1100	455	450	1.11	450/18=25
1500	333	324	2.7	324/18=18
1700	294	288	2.04	288/18=16
2000	250	252	0.8	252/18=14

To Vote For This Design, Circle No. 306

LISTING 1–ROUTINE FOR GENERATING ARBITRARY FREQUENCIES

```
C1 EQU 40    ; Count for 500Hz
C2 EQU 28    ; Count for 700Hz
C3 EQU 20    ; Count for 1000Hz
C4 EQU 19    ; Count for 1050Hz
C5 EQU 18    ; Count for 1100Hz
C6 EQU 14    ; Count for 1500Hz
C7 EQU 12    ; Count for 1700Hz
C8 EQU 10    ; Count for 2000Hz

F1 EQU 21    ; Internal RAM locations for storing the
F2 EQU 22    ; above count values
F3 EQU 23
F4 EQU 24
F5 EQU 25
F6 EQU 26
F7 EQU 27
F8 EQU 28

org 0
ajmp main;

;----------Timer T0 service routine --------------

org 00BH            ; Timer 0 to service vector

cpl p3.4
djnz F1,CF2 ; 24 osc. periods
cpl p1.7    ; 12 osc. periods
mov F1,#C1  ; 24 osc. periods;
          ; Total Osc.cycles = 60 * 8 + 12 = 492
            ; with 24MHz crystal this amounts to
            ; 20.5 microseconds

CF2:
djnz F2,CF3 ; decrement counter for freq. f2
cpl p1.6    ; if it reduces to 0 then complement the
mov F2,#C2  ; logic on the corresponding pin and reload
            ; the count value

CF3:
djnz F3,CF4
cpl p1.5
mov F3,#C3

CF4:
djnz F4,CF5
cpl p1.4
mov F4,#C4

CF5:
djnz F5,CF6
cpl p1.3
mov F5,#C5

CF6:
djnz F6,CF7
cpl p1.2
mov F6,#C6

CF7:
djnz F7,CF8
cpl p1.1
mov F7,#C7

CF8:
djnz F8,exit
cpl p1.0
mov F8,#C8

exit:
reti        ; End of timer interrupt routine

main:
call init;

LOOP: JMP LOOP;


;------ Initialization Routine -----------
init:
        mov F1,#C1      ;  Initialize count values
        mov F2,#C2
        mov F3,#C3
        mov F4,#C4
        mov F5,#C5
        mov F6,#C6
        mov F7,#C7
        mov F8,#C8

; The following code sets timer T0 to 25 microseconds
; and enables the timer intterupt

        mov TMOD, #02H   ; set timer mode to mode 2
        mov tl0,#0ceh    ; (8-bit auto-reload mode)
        mov th0,#0ceh    ; set reload value
        setb ET0         ; enable timer interrupt
        setb EA          ; master interrupt enable
        setb TR0         ; start timer
ret

end
```